A COMPARISON OF CABAC THROUGHPUT FOR HEVC/H.265 VS. AVC/H.264

Vivienne Sze[†], *Madhukar Budagavi*[‡]

[†]Massachusetts Institute of Technology
[‡]Texas Instruments

ABSTRACT

The CABAC entropy coding engine is a well known throughput bottleneck in the AVC/H.264 video codec. It was redesigned to achieve higher throughput for the latest video coding standard HEVC/H.265. Various improvements were made including reduction in context coded bins, reduction in total bins and grouping of bypass bins. This paper discusses and quantifies the impact of these techniques and introduces a new metric called Bjøntegaard delta cycles (BD-cycle) to compare the CABAC throughput of HEVC vs. AVC. BD-cycle uses the Bjøntegaard delta measurement method to compute the average difference between the cycles vs. bit-rate curves of HEVC and AVC. This metric is useful for estimating the throughput of an HEVC CABAC engine from an existing AVC CABAC design for a given bit-rate. Under the common conditions set by the JCT-VC standardization body, HEVC CABAC has an average BD-cycle reduction of 31.1% for all intra, 24.3% for low delay, and 25.9% for random access, when processing up to 8 bypass bins per cycle.

Index Terms— HEVC, H.265, AVC, H.264, CABAC, throughput, arithmetic coding

1. INTRODUCTION

HEVC/H.265 is the latest video compression standard that was finalized in January 2013 by the Joint Collaborative Team on Video Coding (JCT-VC) [1]. HEVC delivers 50% higher coding efficiency than its predecessor AVC/H.264 [2]. In addition, the committee also sought to increase the throughput of various tools in the standard. Enabling higher throughput is desirable since it can be used to support higher pixel rates (for higher resolutions and frame rates), and throughput can also be traded off for reduced power consumption via voltage scaling [3]. Parallelism is an effective way to increase throughput. However, this is inherently challenging as the purpose of video compression is to remove redundancy which introduces dependency and makes parallelism difficult.

This is particularly true for the design of the entropy coding engine. HEVC uses context adaptive binary arithmetic coding (CABAC) to losslessly compress syntax elements to encoded bits. While CABAC provides higher coding efficiency compared to other forms of entropy coding, such as context adaptive variable length coding (CAVLC), it contains tight feedback loops which make it difficult to parallelize. CABAC, which was also used in AVC, is a well known throughput bottleneck in the video codec, particularly at the decoder. Accordingly, multiple core experiments and ad hoc groups were established to study and improve the CABAC throughput throughout the HEVC standardization process [4, 5]. The resulting CABAC for HEVC delivers higher throughput than AVC [6].

This paper will begin by providing an overview of CABAC. It will then highlight some of the techniques that were used to gain higher throughput for HEVC. Finally, it will quantify the impact of these techniques, and compare the overall CABAC throughput in HEVC vs. AVC under various test conditions.

2. OVERVIEW OF CABAC

Entropy coding is used in video coding to losslessly compress syntax elements (e.g. motion vectors, prediction modes, coefficients) to encoded bits. It is the last step in the video encoder, and the first step in the video decoder. CABAC is a form of entropy coding used in both AVC and HEVC. In AVC, two methods of entropy coding are used: CAVLC and CABAC. CABAC provides 9-14% higher coding efficiency than CAVLC [7], however CAVLC provides higher throughput. In HEVC, CABAC is the only entropy coding engine; thus, it must provide both high coding efficiency and high throughput.

CABAC involves three main steps:

1) Binarization: Syntax elements are mapped to binary symbols (bins) using a binarization process. Various forms of binarization are used in AVC and HEVC (e.g. Exp-Golomb, fixed length, truncated unary, custom). Combinations of different binarizations are also allowed where the prefix and suffix are binarized differently. For instance, the prefix can be truncated unary and the suffix can be fixed length (this combination is also referred to as truncated rice). Alternatively, truncated unary can be used for the prefix, and Exp-Golomb for the suffix. The standard defines which type of binarization is used for each syntax element.

2) Arithmetic Coding: The bins are compressed into bits using arithmetic coding (i.e. multiple bins can be represented

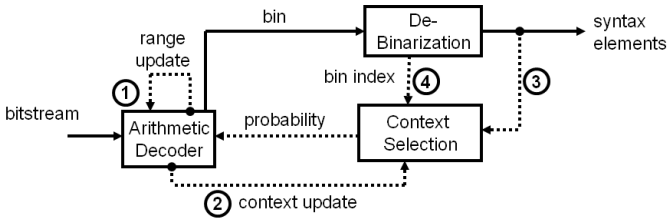

Fig. 1: Three key steps in CABAC: Binarization, Context Selection and Arithmetic Coding. Feedback loops in the decoder are highlighted with dashed lines.

by a single bit); this allows syntax elements to be represented by a fractional number of bits, which improves coding efficiency. Arithmetic coding involves recursive sub-interval division, where a range is divided into two subintervals based on the probability of the symbol that is being compressed. The encoded bits represent an offset that, when converted to a binary fraction, selects one of the two subintervals, which indicates the value of the decoded bin. After every decoded bin, the range is updated to equal the selected subinterval, and the interval division process repeats itself. In order to effectively compress the bins to bits, the probability of the bins must be accurately estimated.

3) Context Selection: The context modeling and selection is used to accurately model the probability of each bin. The probability of the bin depends on the type of syntax element it belongs to, the bin index within the syntax element (e.g. most significant bin or least significant bin) and the properties of spatially neighboring coding units. There are several hundred different context models used in AVC and HEVC. As a result, a large finite state machine is needed to select the correct context for each bin. In addition, the estimated probability of the selected context model is updated after each binary symbol is encoded or decoded.

At the decoder, there are several feedback loops in the CABAC as highlighted in Fig. 1. The context and probability of the next bin depends on the decoded value of the current bin; the current bin determines the bin index and syntax element and consequently the context of the next bin. This bin-to-bin dependency makes it difficult to process multiple bins concurrently. The context and probability for the next bin can be speculatively determined to increase concurrency; however, due to the complexity of the context selection finite state machine, these speculative operations are expensive in terms of power and area, and grow exponentially as the number of concurrent bins increases. Note that the context update and range update feedback loops are simpler than the context selection loops and thus do not affect throughput as severely.

Not all bins are coded using an estimated probability (i.e. context coded). Bins can also be coded assuming equal probability of 0.5 (i.e. bypass coded)[1]. As a result, bypass coded bins avoid the feedback loop for the context selection. In addition, the arithmetic coding is also simpler and faster for bypass coded bins, as the division of the range into subintervals can be done by a shift, rather than a look up table which is required for the context coded bins. Thus multiple bypass bins can be processed concurrently in the same cycle at lower power and area cost than context coded bins. This property is highly leveraged by the throughput improvement techniques describe in the next section.

[1]The standard defines which bins are context coded or bypass coded.

3. THROUGHPUT IMPROVEMENTS IN HEVC

In HEVC, the binarization and context selection of CABAC were modified, while the arithmetic coding engine remained the same as AVC. This section highlights three techniques that were used to improve the throughput of CABAC in HEVC.

3.1. Reduce total number of bins

The binarization of the coefficient level was modified to reduce the total number of bins. The coefficient levels account for a significant portion (on average 15 to 25%) of the total number of bins. While AVC uses truncated unary prefix followed by an Exp-Golomb suffix, HEVC uses a truncated unary prefix followed by fixed length suffix [8]; this combination is also known as truncated rice [9]. Up to the value of 12, HEVC and AVC have the same number of bins; however, for coefficient values above 12, the binarization used in HEVC always results in fewer bins than AVC. The transition point between the prefix and suffix was set such that the maximum total number of bins for coefficient level[2] was reduced from 43 to 34 [10]. The maximum total number of bins for delta QP was also reduced from 53 to 15 by using truncated unary plus Exp-Golomb rather than unary, and signaling the sign value separately [11, 12]. Overall, for a 16x16 block of pixels, the worst case total number of bins was reduced in HEVC by 1.5x compared to AVC [6].

3.2. Reduce number of context coded bins

The number of context coded bins was significantly reduced for syntax elements such as motion vectors and coefficient levels. For these syntax elements, the first few bins (i.e. the most significant bins) were context coded, and the remaining bins were bypass coded. For instance, in AVC, the first 9 bins of the motion vector difference were context coded. For HEVC, it was determined that only the first two bins had to be context coded [13]. Similarly, the number of context coded bins for each coefficient level was reduced from 14 in AVC to either 1 or 2 (depending on the number of coefficients per 4x4 block) in HEVC [14]. Table 1 summarizes the reduction in context coded bins for various syntax elements. Overall, if we account of the number of times each syntax element appears for a 16x16 block of pixels, the worst case number

[2] $coeff\_abs\_level\_greater1\_flag$, $coeff\_abs\_level\_greater2\_flag$, and $coeff\_abs\_level\_remaining$

Syntax element	AVC	HEVC
motion vector difference [13]	9	2
coefficient level [9]	14	1 or 2
reference index [15]	31	2
delta QP [11]	53	5
remainder of intra prediction mode	3	0

Table 1: Number of context coded bins for various syntax elements.

Syntax element	Number of syntax elements
motion vector difference [13]	2
coefficient level [17]	16
coefficient sign [16]	16
remainder of intra prediction mode [18]	4

Table 2: Bypass grouping across syntax elements.

of context coded bins was reduced by over 8x compared to AVC [6].

3.3. Grouping of bypass bins

Due to the effort to reduce the number of context coded bins, bypass bins account for a significant portion of the total bins in HEVC. As a result, processing multiple bypass bins per cycle can significantly increase the overall CABAC throughput. Multiple bypass bins can only be processed in the same cycle if bypass bins appear consecutively in the binstream. Thus long runs of bypass bins result in higher throughput than frequent switching between bypass and context coded bins. Accordingly, bypass bins were grouped together across multiple syntax elements to maximize the run length of bypass bins. For instance, the sign bins of the different coefficients were grouped together [16], and the bypass portion of the coefficient levels were grouped together [17]. Table 2 summarizes the syntax elements where bypass grouping was used.

Note that several other throughput improvements were made to CABAC in HEVC including simplifying the context selection for significance map to reduce the complexity of speculative computations, and reducing the dependency on neighbors for context selection to reduce logic and also line buffer size. These additional improvements are discussed in detail in [6].

4. EXPERIMENT RESULTS

This section quantifies and compares the throughput of CABAC in AVC and HEVC. While previous work [6] compares the CABAC throughput for the *worst* case, this paper focuses on the *average* case. The throughput was measured for 24 video sequences of various resolutions that were used by JCT-VC during the HEVC standardization process [19]. These sequences were encoded with HM-8.0 reference software for HEVC [20], and JM-18.4 reference software for AVC [21]. The HM-8.0 encoder was configured based on the common conditions set by JCT-VC [22], while the JM-18.4 encoder was configured based on the HM-like conditions provided in [23]. Under the common condition, for both AVC and HEVC, 12 encoded bitstreams are generated for each of the 24 video sequences, which covers four quantization points (QP = 22, 27, 32, 37) and three different configurations (all intra, low delay and random access).

To determine the CABAC throughput, we estimated the total number of cycles required to process each of the encoded bitstreams based on the total number of bins, the number of context coded bins, and the run lengths of the bypass bins. For this paper, we computed the CABAC cycle count for CABAC engines that could process one context coded bin per cycle, and between 1 to 16 bypass coded bins per cycle. However, the analysis can be extended to other architectures and software implementations with different cycle times for context coded and bypass coded bins. Decoding multiple bypass bins per cycle was explored in several AVC architectures [24, 25].

The cycle count, PSNR and bit-rate were determined for each of the encoded bitstreams. Data for the four quantization points were then used to generate bit-rate vs. PSNR (rate-distortion) curves and bit-rate vs. cycles curves for each video sequence and configuration. Fig. 2 and Fig. 3 show examples of these curves for the Kimono sequences encoded with the random access configuration. The rate-distortion curve is commonly used to evaluate the coding efficiency (independent of throughput). The cycles vs. bit-rate curve, introduced in this work, can be used to evaluate the throughput for a given bit-rate (independent of PSNR).

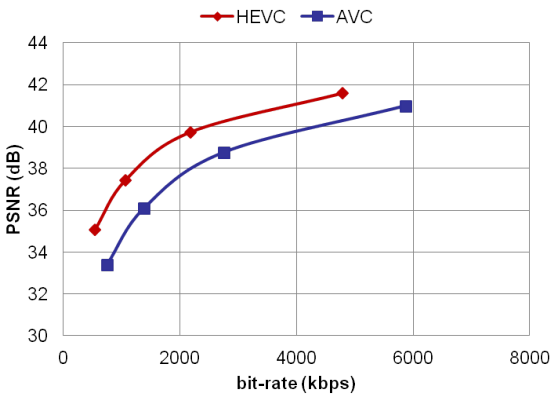

Fig. 2: Rate-Distortion curve for Kimono (Random Access).

The Bjøntegaard Delta [26] measurement method, which computes the average difference between two curves, was extensively used in both the AVC and HEVC standardization process to compare the rate-distortion curves of different tools. This paper extends the use of the Bjøntegaard Delta measurement method to measure the average difference be-

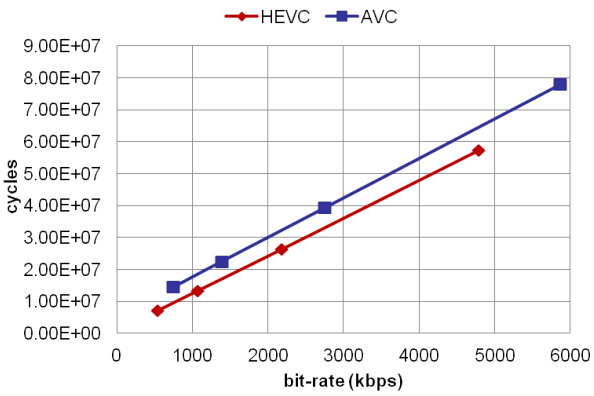

Fig. 3: Cycles-Rate curve for Kimono (Random Access). Cycles computed assuming *one bypass bin per cycle*.

Class (Resolution)	All Intra	Low Delay	Random Access
A (2560×1600)	-7.4%		-14.1%
B (1920×1080)	-5.2%	-7.9%	-9.8%
C (832×480)	-4.4%	-8.2%	-10.2%
D (416×240)	-5.2%	-5.7%	-8.1%
E (1280×720)	-7.1%	-23.3%	
F (1280×720)	-12.6%	-23.5%	-20.5%
All	**-6.9%**	**-12.9%**	**-10.5%**

Table 3: BD-cycle reduction for cycles vs. bit-rate for *one bypass bin per cycle*.

tween the cycles vs. bit-rate curves for AVC and HEVC. The metric BD-cycle measures the difference in cycles between the two curves averaged across bit-rate. This metric is useful for estimating the throughput of an HEVC CABAC engine from an existing AVC CABAC design for a given bit-rate.

4.1. Impact of reducing total number of bins

The BD-cycle reduction was computed for each of the video sequences and configurations. Table 3 shows BD-cycle reduction of HEVC over AVC when processing one bypass bin per cycle. In this case, the cycle count estimate is equal to the total number of bins. Thus, the BD-cycle reduction of 6.9% for all intra, 12.9% for low delay and 10.5% for random access is due to the reduction in total number of bins in HEVC. Fig. 3 shows the corresponding bit-rate vs. cycles trade-off curve for the Kimono sequence.

4.2. Impact of reducing number of context coded bins

Table 4, Table 5, and Table 6 show that the BD-cycle reduction increases as the number of bypass bins per cycles increases for the different configurations. The first column of the tables shows the impact of increasing number of bypass bins per cycle for AVC; the AVC encoded bitstream with one bypass bin per cycle is used as the anchor for the BD-cycle calculation. In AVC, processing 16 bypass bins per cycle reduces the BD-cycle by 0.2% for all intra, 1.6% for low delay, and 2.2% for random access.

For HEVC, the number of context coded bins is reduced compared to AVC as shown in Table 7. Accordingly, processing more bypass bins per cycle has a greater impact on BD-cycle reduction for HEVC than AVC. The second column of Table 4, Table 5, and Table 6 shows the BD-cycle reduction achieved in HEVC due to reduction in context coded bins and total bins; the AVC encoded bitstream with the same number of bypass bins per cycle as the HEVC encoded bitstream is used as the anchor for the BD-cycle calculation. Reducing the number of context coded bins and total bins provides an additional BD-cycle reduction of 23.4% for all intra, 19.1% for low delay, and 19.4% for random access, when 16 bypass bins are processed per cycle.

Number of bins per cycle	AVC	Impact of reducing context coded bins and total bins	+ Impact of bypass grouping
1	-0.0%	-6.9%	-6.9%
2	-0.1%	-16.7%	-21.1%
4	-0.2%	-21.3%	-27.8%
8	-0.2%	-23.0%	-31.1%
16	-0.2%	-23.4%	-32.5%

Table 4: Average BD-cycle reduction averaged across bit-rate for various bypass bins per cycles (All Intra).

Number of bins per cycle	AVC	Impact of reducing context coded bins and total bins	+ Impact of bypass grouping
1	-0.0%	-12.9%	-12.9%
2	-0.8%	-16.7%	-19.8%
4	-1.3%	-18.5%	-23.0%
8	-1.6%	-18.9%	-24.3%
16	-1.6%	-19.1%	-24.8%

Table 5: Average BD-cycle reduction averaged across bit-rate for various bypass bins per cycles (Low Delay).

Number of bins per cycle	AVC	Impact of reducing context coded bins and total bins	+ Impact of bypass grouping
1	-0.0%	-10.5%	-10.5%
2	-1.1%	-15.9%	-19.7%
4	-1.7%	-18.5%	-24.0%
8	-2.1%	-19.3%	-25.9%
16	-2.2%	-19.4%	-26.7%

Table 6: Average BD-cycle reduction averaged across bit-rate for various bypass bins per cycles (Random Access).

	All Intra	Low Delay	Random Access
AVC	84.9%	86.7%	85.3%
HEVC	68.0%	78.2%	73.1%

Table 7: Average percentage of context coded bins.

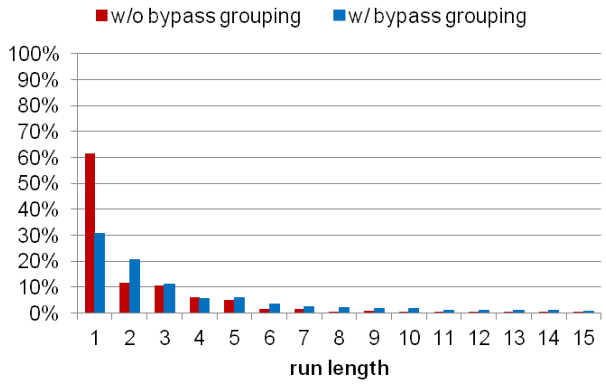

Fig. 4: Distribution of bypass runs with and without bypass grouping for Kimono sequence (Random Access).

4.3. Impact of grouping bypass bins

Finally, bypass grouping was used to achieve longer runs of bypass bins (as shown in Fig. 4) in order to maximize the impact of processing multiple bypass bins per cycle. The third column of Table 4, Table 5, and Table 6 shows the BD-cycle reduction when bypass grouping is used relative to AVC for different number of bypass bins per cycle; again the AVC encoded bitstream with the same number of bypass bins per cycle as the HEVC encoded bitstream is used as the anchor for the BD-cycle calculation. In HEVC, processing 16 bypass bins per cycle reduces the BD-cycle by 32.5% for all intra, 24.8% for low delay, and 27.0% for random access. Thus bypass grouping provides an additional BD-cycle reduction of 9.1% for all intra, 5.7% for low delay, and 7.3% for random access.

Note that the gains of multiple bypass bins per cycle diminish as the number of bins increase beyond 4 to 8 bins per cycle. As increasing the number of bins processed per cycle results in increased area cost, it may not be beneficial to design a CABAC engine that processes more than 8 bypass bins per cycle. Fig. 5 shows the cycles vs. bit-rate curves for 8 bypass bins per cycle and one context coded bin per cycle. Table 8 shows the total BD-cycle reduction of HEVC over AVC for 8 bypass bins per cycle.

5. CONCLUSION

The CABAC entropy coding used in HEVC was designed to deliver higher throughput than AVC. Several techniques were used to improve the throughput including reducing context coded bins, reducing total bins, and grouping bypass bins.

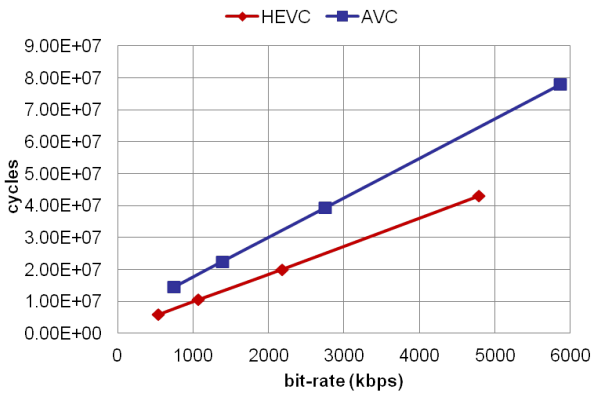

Fig. 5: Cycles-Rate curve for Kimono (Random Access). Cycles computed assuming up to *8 bypass bins per cycle*.

Class	All Intra	Low Delay	Random Access
A	-33.9%		-30.1%
B	-27.7%	-19.2%	-24.3%
C	-28.0%	-20.8%	-26.0%
D	-29.7%	-17.1%	-23.6%
E	-29.3%	-30.4%	
F	-38.3%	-36.8%	-38.9%
All	**-31.1%**	**-24.3%**	**-25.9%**

Table 8: BD-cycle reduction for cycles vs. bit-rate assuming up to *8 bypass bins per cycle*.

These techniques give HEVC up to 31.1% BD-cycle reduction over AVC under common conditions. It should be noted that in the worst case, where bypass bins account for over 90% of the total bins, cycle reduction of up to 50% is achieved [6]. In addition, it was shown that HEVC throughput can be increased significantly by processing more bypass bins per cycle. Based on the analysis, 4 to 8 bypass bins per cycle is likely to provide the best trade-off in terms of throughput improvement vs. area cost. Additional techniques such as memory reduction, reduction of context selection dependencies and parsing dependencies will further reduce cycles and implementation costs to achieve a high throughput CABAC implementation.

6. APPENDIX

The Bjøntegaard Delta measurement method can also be used to evaluate the difference between cycles vs. PSNR curves for HEVC and AVC as shown in Fig. 6. BD-cycle in this case would be a measure of the difference in cycles averaged across PSNR. Table 9 shows that an average BD-cycle reduction of 45.3% for all intra, 52.4% for low delay, and 51.3% for random access is achieved across PSNR. The BD-cycle reduction averaged across PSNR is greater than the BD-cycle reduction averaged across bit-rate since it also accounts for

the fact that for the same PSNR, HEVC has lower bit-rate than AVC due to improved prediction, larger block and transform sizes, and subsequently lower residual. For instance, for the Kimono sequence (1080p@ 24fps) shown in Fig. 2, Fig. 8 and Fig. 6, with similar PSNR of 40dB, HEVC requires half the number of cycles as AVC (22 Mcycles vs. 58 Mcycles). This can be attributed to reduction in bit-rate due to improved coding efficiency (2.4 Mbps vs. 4.4 Mbps) as well as reduction in context coded bins (68% vs. 84%).

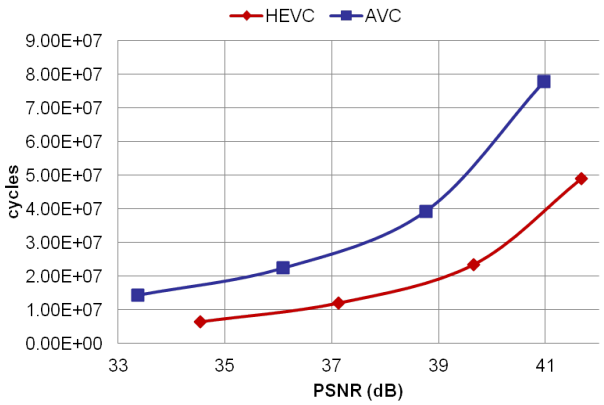

Fig. 6: Cycles-Distortion Curve for Kimono (Random Access). Cycles computed assuming up to *8 bypass bins per cycle*.

Class	All Intra	Low Delay	Random Access
A	-47.9%		-55.6%
B	-42.7%	-54.7%	-55.6%
C	-40.7%	-47.0%	-48.1%
D	-40.1%	-42.6%	-44.8%
E	-47.4%	-63.4%	
F	-53.8%	-56.3%	-56.5%
All	**-45.3%**	**-52.4%**	**-51.3%**

Table 9: BD-cycle reduction for cycles vs. PSNR assuming up to *8 bypass bins per cycle*.

7. REFERENCES

[1] B. Bross, W. J. Han, G. Sullivan, J.-R. Ohm, Y. K. Wang, and T. Wiegand, "High Efficiency Video Coding (HEVC) text specification draft 10," Jan. 2013.

[2] G.J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. on CSVT*, vol. 22, no. 12, pp. 1649–1668, 2012.

[3] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, April 1992.

[4] M. Budagavi, G. Martin-Cocher, and A. Segall, "JCTVC-D009:JCT-VC AHG report: Entropy coding," Joint Collaborative Team on Video Coding, March 2010.

[5] V. Sze, K. Panusopone, J. Chen, T. Nguyen, and M. Coban, "JCTVC-C511: Description of Core Experiment 11: Coefficient Scanning and Coding," Joint Collaborative Team on Video Coding, Oct. 2010.

[6] V. Sze and M. Budagavi, "High Throughput CABAC Entropy Coding in HEVC," *IEEE Trans. on CSVT*, vol. 22, no. 12, pp. 1778–1791, 2012.

[7] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. on CSVT*, vol. 13, no. 7, pp. 620– 636, July 2003.

[8] W.-J. Chien, M. Karczewicz, J. Sole, and J. Chen, "JCTVC-I0487: On coefficient level remaining coding," Joint Collaborative Team on Video Coding, April 2012.

[9] T. Nguyen, "JCTVC-E253: CE11: Coding of transform coefficient levels with Golomb-Rice codes," Joint Collaborative Team on Video Coding, March 2011.

[10] M. Budagavi and V. Sze, "JCTVC-J0142: coeff_abs_level_remaining maximum codeword length reduction.," Joint Collaborative Team on Video Coding, July 2012.

[11] V. Sze, M. Budagavi, V. Seregin, J. Sole, and M. Karczewicz, "JCTVC-J0089: AHG5: Bin reduction for delta QP coding," Joint Collaborative Team on Video Coding, July 2012.

[12] K. Chono and H. Aoki, "JCTVC-F046: Efficient binary representation of cu_qp_delta syntax for CABAC," Joint Collaborative Team on Video Coding, July 2011.

[13] T. Nguyen, D. Marpe, H. Schwarz, and T. Wiegand, "JCTVC-F455: Modified binarization and coding of MVD for PIPE/CABAC," Joint Collaborative Team on Video Coding, June 2011.

[14] J. Chen, W. J. Chien, R. Joshi, J. Sole, and M. Karczewicz, "JCTVC-H0554: Non-CE1: throughput improvement on CABAC coefficients level coding," Joint Collaborative Team on Video Coding, Feb. 2012.

[15] V. Seregin, J. Sole, M. Karczewicz, X. Wang, V. Sze, and M. Budagavi, "JCTVC-J0098: AHG5: Bypass bins for reference index coding," Joint Collaborative Team on Video Coding, July 2012.

[16] M. Budagavi, "JCTVC-C062: TE8: TI parallel context processing (PCP) proposal," Joint Collaborative Team on Video Coding, Oct. 2010.

[17] V. Sze and M. Budagavi, "JCTVC-F130: Parallel Context Processing of Coefficient Level," Joint Collaborative Team on Video Coding, July 2011.

[18] W.-J. Chien, J. Chen, M. Coban, and M. Karczewicz, "JCTVC-I0302: Intra mode coding for INTRA_NxN," Joint Collaborative Team on Video Coding, April 2012.

[19] J. Ohm, G.J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards - Including High Efficiency Video Coding (HEVC)," *IEEE Trans. on CSVT*, vol. 22, no. 12, pp. 1669–1684, 2012.

[20] "HEVC Test Model, HM 8.0," https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-8.0/.

[21] "H.264/AVC Reference Software, JM 18.4," http://iphome.hhi.de/suehring/tml/.

[22] F. Bossen, "JCTVC-J1100: HM 8 Common Test Conditions and Software Reference Configurations," Joint Collaborative Team on Video Coding, July 2012.

[23] B. Li, G. J. Sullivan, and J. Xu, "JCTVC-M0329: Comparison of Compression Performance of HEVC Draft 10 with AVC High Profile," Joint Collaborative Team on Video Coding, April 2013.

[24] B. Shi, W. Zheng, H. S. Lee, D. X. Li, and M. Zhang, "Pipelined Architecture Design of H.264/AVC CABAC Real-Time Decoding," in *IEEE Inter. Conf. on Circuits and Systems for Communications*, May 2008, pp. 492 –496.

[25] Y. C. Yang and J. I. Guo, "High-Throughput H.264/AVC High-Profile CABAC Decoder for HDTV Applications," *IEEE Trans. on CSVT*, vol. 19, no. 9, pp. 1395 –1399, September 2009.

[26] G. Bjøntegaard, "VCEG-M33: Calculation of Average PSNR Differences between RD curves," Video Coding Experts Group (VCEG), April 2001.